

Introduction to MLflow Projects

INTRODUCTION TO MLFLOW



Weston Bassler
Senior MLOps Engineer

MLflow Projects

- Reproducible
- Reusable
- Portable



¹ unsplash.com

MLproject

```
project/  
  MLproject  
  train_model.py  
  python_env.yaml  
  requirements.txt
```



¹ github.com

MLproject file

- `name:` - Name of the Project
- `entry_points:`
 - Command(s) to run
 - `.py` and `.sh` files
 - Workflows
- `python_env:`
 - Python environment
 - `conda.yaml` or `python_env.yaml`



¹ wikipedia.org

MLproject example

```
name: salary_model
entry_points:
  main:
    command: "python train_model.py"
python_env: python_env.yaml
```

train_model.py

```
# Import libraries and modules
import mlflow
import mlflow.sklearn
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Training Data
df = pd.read_csv('Salary_predict.csv')
X = df[["experience", "age", "interview_score"]]
y = df[["Salary"]]
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=0)
```

train_model.py

```
# Set Auto logging for Scikit-learn flavor  
mlflow.sklearn.autolog()  
  
# Train the model  
lr = LinearRegression()  
lr.fit(X_train, y_train)
```

python_env.yaml

```
python: 3.10.8
build_dependencies:
- pip
- setuptools
- wheel
dependencies:
- -r requirements.txt
```

requirements.txt

mlflow

scikit-learn

Let's practice!

INTRODUCTION TO MLFLOW

Running MLflow Projects

INTRODUCTION TO MLFLOW



Weston Bassler

Senior MLOps Engineer

API and command line



Projects API

```
mlflow.projects
```

```
mlflow.projects.run()
```

- `uri` - URI to MLproject file
- `entry_point` - Entry point to start run from MLproject
- `experiment_name` - Experiment to track training run
- `env_manager` - Python environment manager: `local`, `virtualenv`, or `conda`

```
# Run MLflow Project
mlflow.projects.run(
    uri='./',
    entry_point='main',
    experiment_name='My Experiment',
    env_manager='conda'
)
```

MLproject

```
name: salary_model
python_env: python_env.yaml
entry_points:
  main:
    command: "python train_model.py"
```

train_model.py

```
# Import libraries and modules
import mlflow
import mlflow.sklearn
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Training Data
df = pd.read_csv('Salary_predict.csv')
X = df[['experience", "age", "interview_score']]
y = df[['Salary']]
```

train_model.py

```
# Train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,
                                                    random_state=0)

# Set Auto logging for Scikit-learn flavor
mlflow.sklearn.autolog()

# Train model
lr = LinearRegression()
lr.fit(X_train, y_train)
```

Projects run

```
# Import MLflow Module
import mlflow

# Run local Project
mlflow.projects.run(uri='./', entry_point='main',
                     experiment_name='Salary Model')
```

Run output

```
# Run Local Project  
mlflow.projects.run(uri='./', entry_point='main',  
                     experiment_name='Salary Model')
```

```
2023/04/02 14:33:23 INFO mlflow.projects: 'Salary Model' does not exist.  
Creating a new experiment  
2023/04/02 14:33:23 INFO mlflow.utils.virtualenv: Installing python 3.10.8 if it  
does not exist  
2023/04/02 14:33:23 INFO mlflow.utils.virtualenv: Creating a new environment  
.mlflow/envs/mlflow-44f5094bba686a8d4a5c772  
created virtual environment CPython3.10.8.final.0-64 in 236ms  
2023/04/02 14:33:23 INFO mlflow.utils.virtualenv: Installing dependencies
```

Run output

```
2023/04/02 14:33:59 INFO mlflow.projects.backend.local: === Running command  
'source /.mlflow/envs/mlflow-44f5094bba686a8d4a5c772/bin/activate && python  
train_model.py' in run with ID '562916d45aeb48ec84c1c393d6e3f5b6' ===  
2023/04/02 14:34:34 INFO mlflow.projects: === Run (ID  
'562916d45aeb48ec84c1c393d6e3f5b6') succeeded ===
```

MLflow Tracking

› Parameters (4)

› Metrics (5)

› Tags (2)

▼ Artifacts

The screenshot shows the MLflow UI interface. On the left, there is a sidebar with a tree view under the 'model' folder. The files listed are: MLmodel, conda.yaml, model.pkl, python_env.yaml, requirements.txt, and estimator.html. To the right of the sidebar, the 'Full Path' is displayed as 'Full Path:/mlruns/5/562916d45aeb48ec...'. Below this, there is a blue button labeled 'Register Model' with a clipboard icon. The main content area is titled 'MLflow Model'. It contains text explaining how to make predictions using the logged model and provides a link to 'register to the model registry'.

Full Path:/mlruns/5/562916d45aeb48ec... [Register Model](#)

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also [register to the model registry](#) to version control

Command line

```
mlflow run
```

- `--entry-point` - Entry point to start run from MLproject
- `--experiment-name` - Experiment to track training run
- `--env-manager` - Python environment manager: `local` , `virtualenv` , or `conda`
- `URI` - URI to MLproject file

Run command

```
# Run main entry point from Salary Model experiment  
mlflow run --entry-point main --experiment-name "Salary Model" ./
```

```
2023/04/02 15:23:34 INFO mlflow.utils.virtualenv: Installing python 3.10.8 if it  
does not exist  
2023/04/02 15:23:34 INFO mlflow.utils.virtualenv: Environment  
.mlflow/envs/mlflow-44f5094bba686a8d4a5c772 already exists  
2023/04/02 15:23:34 INFO mlflow.projects.backend.local: === Running command 'source  
/Users/weston/.mlflow/envs/mlflow-44f5094bba686a8d4a5c772/bin/activate && python  
train_model.py' in run with ID 'da5b37b6f53245e5bca59ba8ed6d7dc1' ===  
2023/04/02 15:23:38 INFO mlflow.projects: === Run  
(ID 'da5b37b6f53245e5bca59ba8ed6d7dc1') succeeded ===
```

Let's practice!

INTRODUCTION TO MLFLOW

Specifying parameters

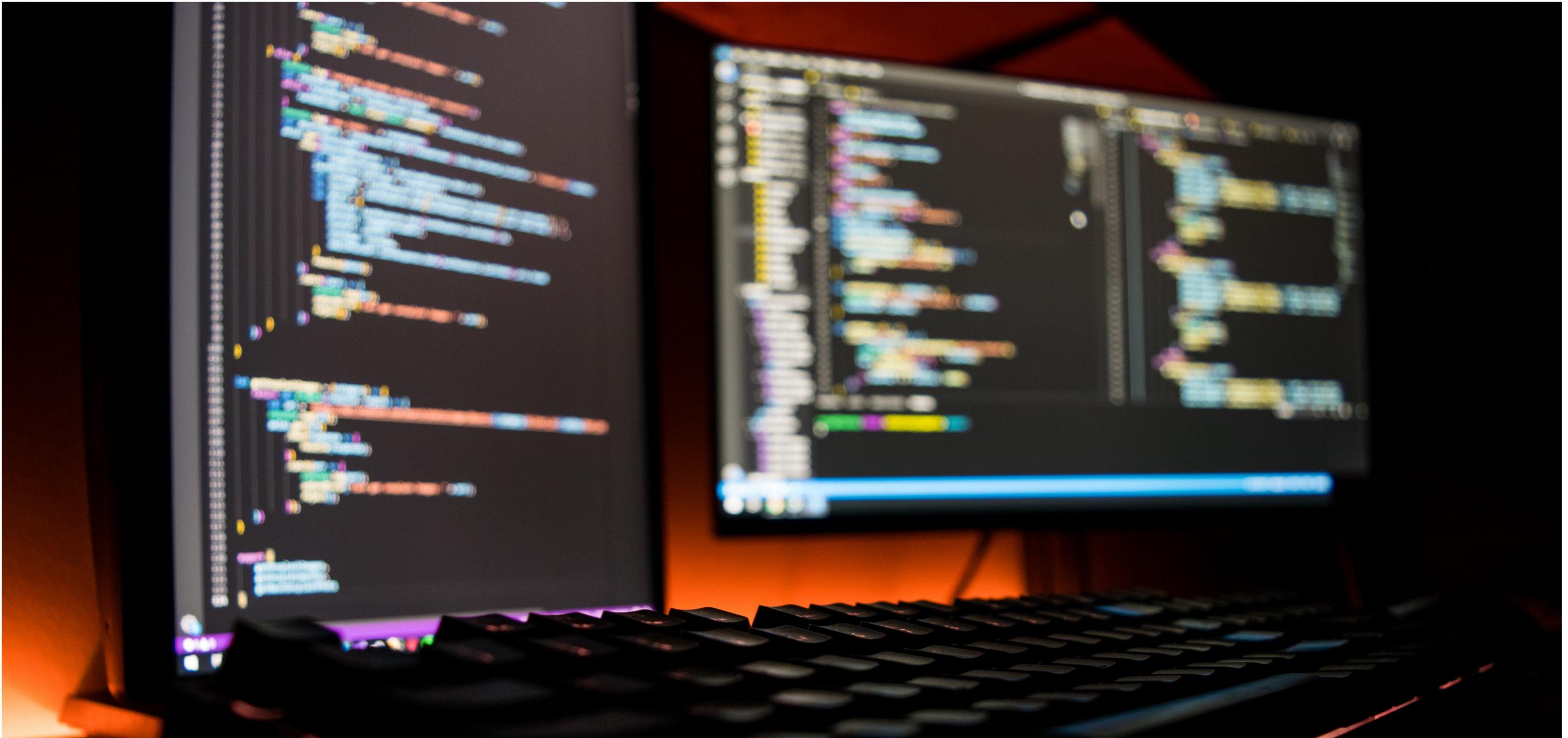
INTRODUCTION TO MLFLOW



Weston Bassler

Senior MLOps Engineer

Parameters



¹ unsplash.com

Specifying parameters

```
parameter_1_name:  
    type: data_type  
    default: default_value  
parameter_2_name:  
    type: data_type  
    default: default_value
```

Specifying parameters

Data types

- float, string, etc...
- defaults to string

Default value

- parameter value if nothing specified during run
- True, False, None, etc...

Parameters block

```
name: project_name
conda_env: python_env.yaml
entry_points:
  main:
    parameters:
      parameter_1:
        type: data_type
        default: default_value
      parameter_2:
        type: data_type
        default: default_value
command: "python train.py {parameter_1_name} {parameter_2_name}"
```

train_model.py

```
# Import libraries and modules
import mlflow
import mlflow.sklearn
import pandas as pd
import sys
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Training Data
df = pd.read_csv('Salary_predict.csv')
X = df[['experience", "age", "interview_score']]
y = df[['Salary']]
```

train_model.py

```
# Train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,
                                                    random_state=0)

# Set Auto logging for Scikit-learn flavor
mlflow.sklearn.autolog()

# Parameters
n_jobs_param = int(sys.argv[1])
fit_intercept_param = bool(sys.argv[2])

# Train model
lr = LinearRegression(n_jobs=n_jobs_param, fit_intercept=fit_intercept_param)
lr.fit(X_train, y_train)
```

MLproject

```
name: salary_model
python_env: python_env.yaml
entry_points:
  main:
    parameters:
      n_jobs_param:
        type: int
        default: 1
      fit_intercept_param:
        type: bool
        default: True
command: "python train_model.py {n_jobs_param} {fit_intercept_param}"
```

Running parameters

Python

```
mlflow.projects.run()
```

```
parameters={  
    'parameter_1': data_value,  
    'parameter_2': data_value,  
}
```

CLI

```
mlflow run
```

```
-P parameter_1=parameter_1_value
```

```
-P parameter_2=parameter_2_value
```

Projects run

```
# Import MLflow Module
import mlflow

# Run local Project
mlflow.projects.run(
    uri='./', entry_point='main',
    experiment_name='Salary Model',
    parameters={
        'n_jobs_param': 2,
        'fit_intercept_param': False
})
```

Output

```
2023/04/07 18:50:22 INFO mlflow.projects.backend.local: === Running command 'source  
/.mlflow/envs/mlflow/bin/activate && python train_model.py 2 False' in run with ID  
'422a6f589c984049ac03698efec8a286' ===  
2023/04/07 18:50:28 INFO mlflow.projects: === Run (ID  
'422a6f589c984049ac03698efec8a286') succeeded ===
```

Run command

```
# Run main entry point from Salary Model experiment  
mlflow run --entry-point main --experiment-name "Salary Model" \  
-P n_jobs_param=3 -P fit_intercept_param=True ./
```

Output

```
# Run main entry point from Salary Model experiment
mlflow run --entry-point main --experiment-name "Salary Model" \
-P n_jobs_param=3 -P fit_intercept_param=True ./
```

```
2023/04/07 19:08:51 INFO mlflow.projects.backend.local: === Running command 'source
/.mlflow/envs/mlflow/bin/activate && python train_model.py 3 True' in run with ID
'7c0a8885a3c7475a9aafc43c6fcae04d' ===
```

Let's practice!

INTRODUCTION TO MLFLOW

Workflows

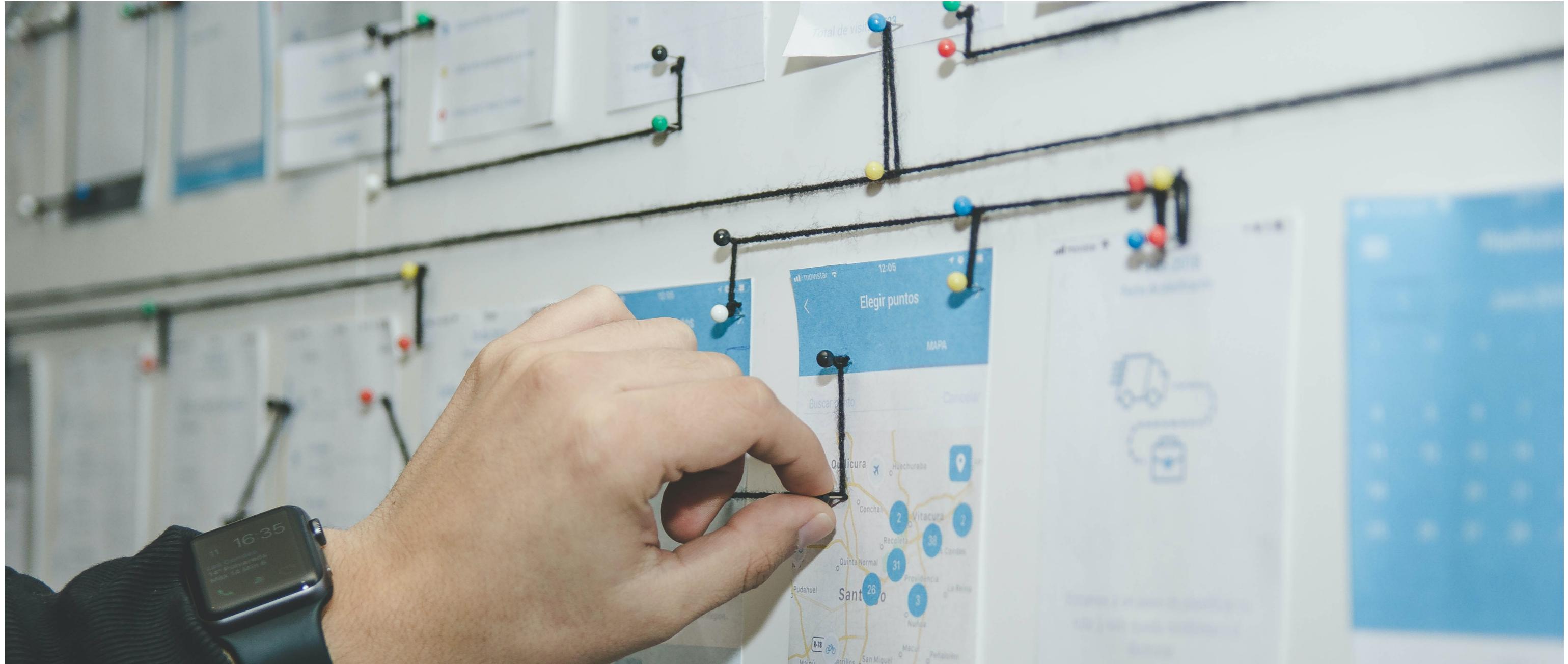
INTRODUCTION TO MLFLOW



Weston Bassler

Senior MLOps Engineer

MLflow Projects



¹ unsplash.com

MLproject

```
name: project_name
python_env: python_env.yaml
entry_points:
  step_1:
    command: "python train_model.py"
  step_2:
    command: "python evaluate_model.py {run_id}"
parameters:
  run_id:
    type: str
    default: None
```

Workflows

```
import mlflow

# Step 1
step_1 = mlflow.projects.run(
    uri='./',
    entry_point='step_1'
)

# Step 2
step_2 = mlflow.projects.run(
    uri='./',
    entry_point='step_2'
)
```

Projects run

```
import mlflow

# Step 1
step_1 = mlflow.projects.run(
    uri='./',
    entry_point='step_1'
)

print(step_1)
```

```
<mlflow.projects.submitted_run.LocalSubmittedRun object at 0x125eac8b0>
```

Projects run

`step_1.cancel()` - Terminate a current run

`step_1.get_status()` - Get the status of a run

`step_1.run_id` - `run_id` of the run

`step_1.wait()` - Wait for the run to finish

Projects run

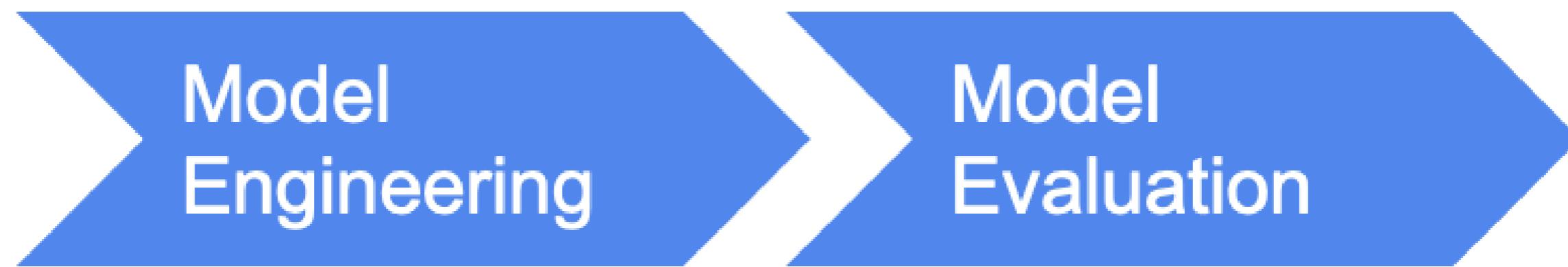
```
import mlflow

# Step 1
step_1 = mlflow.projects.run(
    uri='./',
    entry_point='step_1'
)

# Set variable for step_1 run_id
step_1_run_id = step_1.run_id
```

```
# Step 2
step_2 = mlflow.projects.run(
    uri='./',
    entry_point='step_2',
    parameters={
        'run_id': step_1_run_id
    }
)
```

ML Lifecycle



Let's practice!

INTRODUCTION TO MLFLOW

Wrap-up

INTRODUCTION TO MLFLOW



Weston Bassler

Senior MLOps Engineer

MLflow Components

MLflow Tracking

Record and query experiments: code, data, config, and results

[Read more](#)

MLflow Projects

Package data science code in a format to reproduce runs on any platform

[Read more](#)

MLflow Models

Deploy machine learning models in diverse serving environments

[Read more](#)

Model Registry

Store, annotate, discover, and manage models in a central repository

[Read more](#)

¹ mlflow.org

Chapter 1 - MLflow Tracking

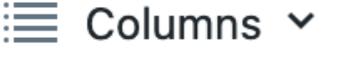
LR Experiment

[Share](#)

Experiment ID: 37 Artifact Location: ./mlruns/37

Description [Edit](#)

Q metrics.rmse < 1 and params.model = "tree"  Sort: Created   Refresh

Columns 

Time created: All time  State: Active  Showing 1 matching run

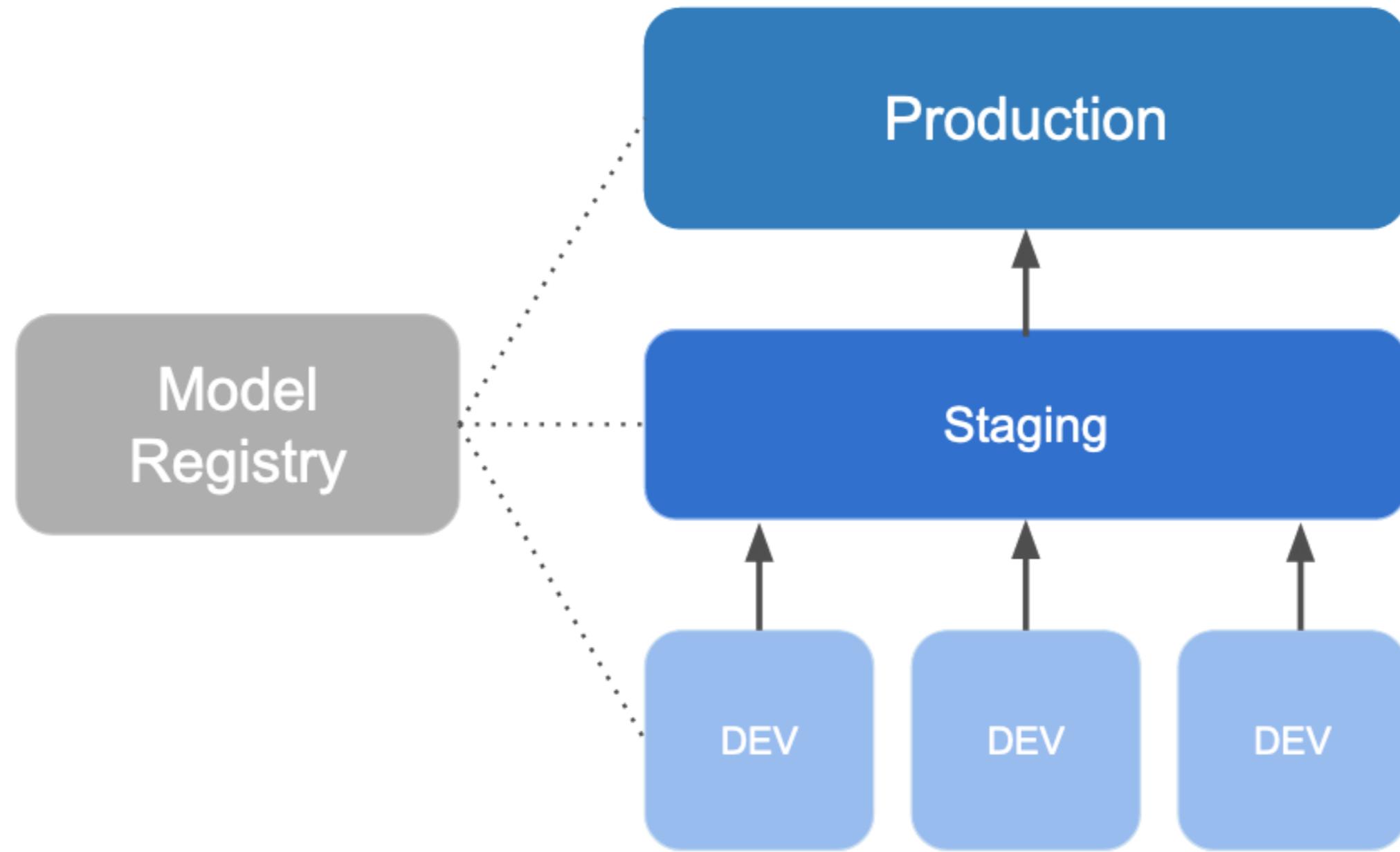
	Run Name	Created	Models	Metrics	Parameters
	Run Name	Created	Models	Metrics	Parameters
<input type="checkbox"/>	silent-slug-662	1 minute ago	-	score	n_jobs
<input checked="" type="checkbox"/>	silent-slug-662	1 minute ago	-	0.951	1

Chapter 2 - MLflow Models

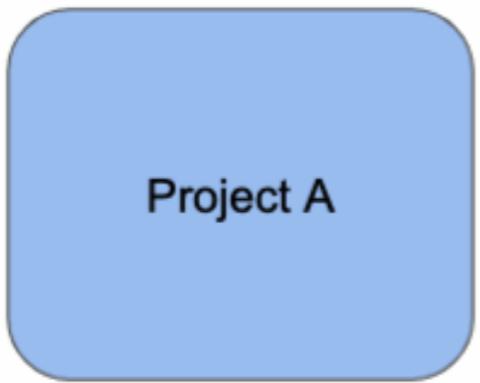
The screenshot shows the MLflow UI interface. On the left, there is a file tree under a 'model' folder. Inside 'model', there is an 'MLmodel' file (which is highlighted with a gray background), a 'conda.yaml' file, a 'model.pkl' file, a 'python_env.yaml' file, a 'requirements.txt' file, and an 'estimator.html' file. Two large red arrows point from the 'python_env.yaml' file and the 'MLmodel' file towards a detailed view on the right. This view displays the contents of the 'MLmodel' file as a YAML configuration. The configuration includes:

```
artifact_path: model
flavors:
  python_function:
    env:
      conda: conda.yaml
      virtualenv: python_env.yaml
    loader_module: mlflow.sklearn
    model_path: model.pkl
    predict_fn: predict
    python_version: 3.10.8
  sklearn:
    code: null
    pickled_model: model.pkl
    serialization_format: cloudpickle
    sklearn_version: 1.1.3
```

Chapter 3 - Model Registry



Chapter 4 - MLflow Projects



Congratulations!



¹ giphy.com

Thank you!

INTRODUCTION TO MLFLOW